Machine Learning
Data Challenge

Gianfranco Salocchi
3047489

Title: Predicting Home Prices in Milano: Data Cleaning and Model Development

Introduction

The objective of this challenge is to construct a model capable of accurately predicting home prices in Milano based on a set of covariates. Covariates, or features, are variables that are believed to influence the outcome, in this case, home prices.

Part 1: Data Cleaning

In the initial phase of the challenge, our primary task was to clean and organize the dataset. Many of the variables initially comprised strings or lists, needing proper transformation for compatibility with the models.

Both the train and test sets included the following covariates:

1.**Square Meters**: Representing the size of the home.

2. **Lift:** Indicating whether the building had a lift, with values originally denoted as 'yes' and 'no', subsequently transformed into '1' and '0', respectively.

3. **Number of Bathrooms**: Reflecting the count of bathrooms in the house.

4. **Room Number**: Signifying the number of rooms in the house.

5. **Other Features**: A list of additional elements present in the house, such as a pool or a garden. For this category, each distinct feature was isolated, and dummy columns were created for each.

6. **Total Floors in Building**: Indicating the number of floors in the building.

7. **Availability**: Differentiating between whether the house was ready for occupancy by the buyer or still under construction. Dates were converted into numerical values using the formula `'time.mktime(datetime.datetime.strptime(s, “%d/%m/%Y”).timetuple())'`, with the rationale that a higher numerical value corresponded to a longer waiting period for the customer.

8. **Condominium Fees.**

9. **Year of Construction:** Representing the year in which the house was built.

10. **Conditions:** Describing the overall condition of the house, with labels transformed into a numerical scale from '1' to '4', with '4' indicating the best condition.

11. **Zone:** Designating the zone of the city. Each zone was one-hot encoded, resulting in a dummy variable for each zone indicating whether the house was located within it.

12.**Floor:** Indicating the floor on which the house was situated.

13. **Centralized Heating**: Differentiating whether the heating system was independent or centralized, with labels transformed into '0' and ‘1'.

14. **Energy Efficiency Class**: Providing a code indicating the energy efficiency of the house. An ordinal scale from '0' to '8' was created, with '8' representing the most efficient label.

15.**Car Parking:** Split into 'private parking' and 'shared parking', with regular expressions employed to extract information and assign the number of parking spots to each variable.

Part 2: Handling Missing Values and Outliers

Having defined the methodology for preprocessing the data, the next step involved addressing missing values and outliers present in the dataset.

Handling Missing Values:

1. **Square Meters:** Some entries reported unrealistically low values, such as 1 or 10 square meters, likely due to data collection errors. Observations with such discrepancies were removed from the training set. In the test set, values were replaced manually after cross-checking with other covariates.

2. **Lift:** Null values for this variable were substituted with '0', indicating the absence of a lift.

3. **Bathrooms Number:** Null values were replaced with '1', assuming each house would have at least one bathroom.

4. **Other Features:** Dummy variables corresponding to other features were set to '0' for observations with null values for this variable.

5. **Total Floors in Building:** Null values were replaced with '1', as every structure is expected to have at least one floor.

6. **Availability:** Null values were set to '0', under the assumption that most houses are available immediately.

7. **Condominium Fees:** While setting null values to '0' felt inappropriate as most apartments have fees, null values were replaced with '0'. Additionally, a dummy variable was created to isolate these cases, along with an interaction variable (the product of the null values dummy and the fees variable).

8. **Year of Construction:** Null values were replaced with the most common year of construction, 1960.

9. **Condition:** Treated as a categorical variable, with a '—' category indicating no condition. This was later one-hot encoded.

10. **Zone:** To ensure compatibility with the test data, zones absent in the test set were removed from the train data. For observations in removed zones or with null values, a 'no-zone' dummy variable was set.

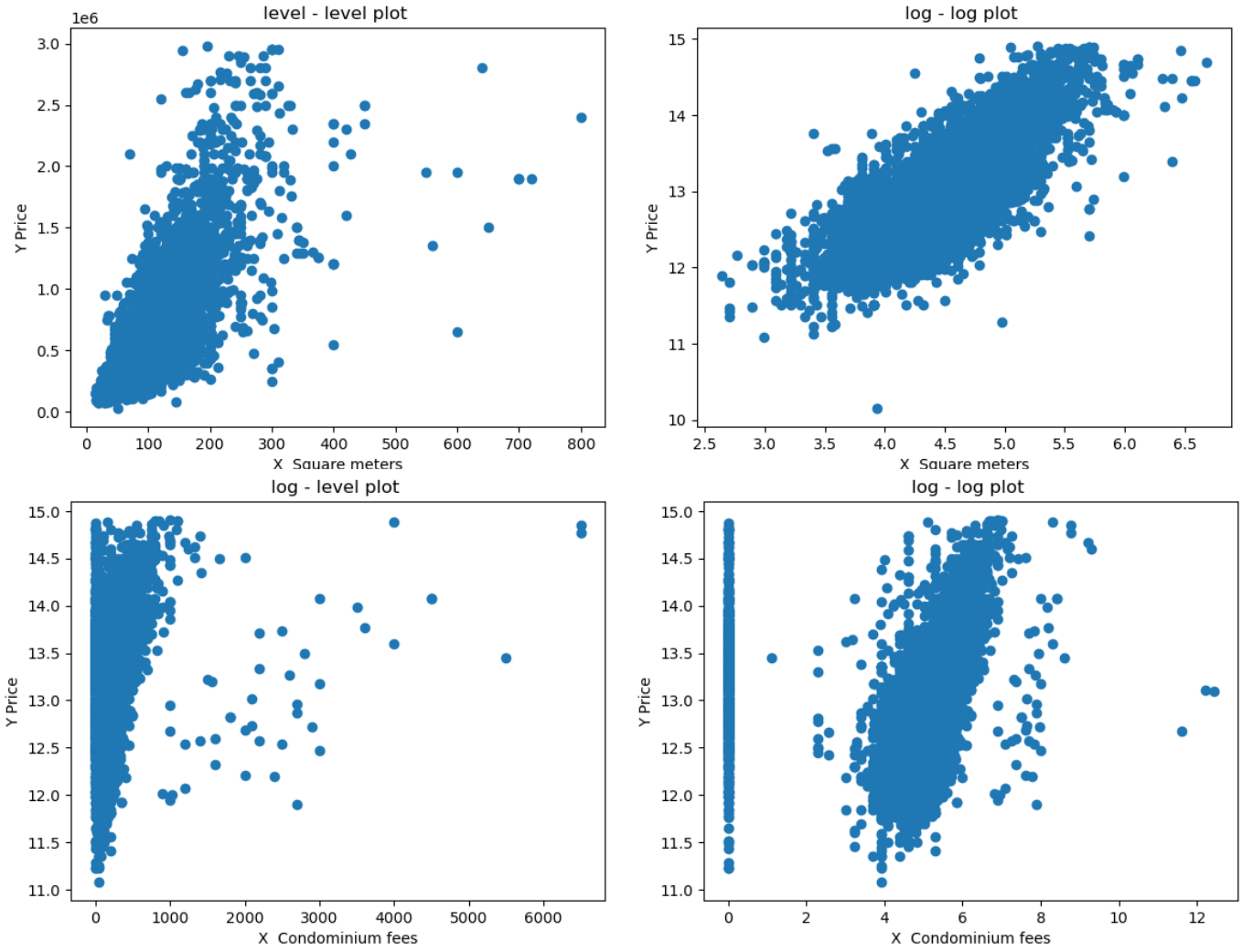
11. **Heating Centralized:** Null values were set to '0', assuming separate units in independent houses.

12. **Energy Efficiency Class:** Null values were replaced with the middle energy class.

Part 3: Data Exploration

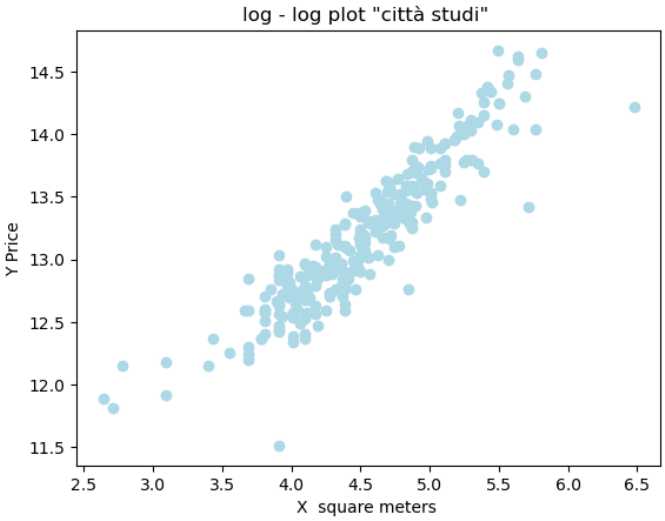
With a clean dataset at hand, the next step involved exploring the relationships between various covariates and home prices through visualization.

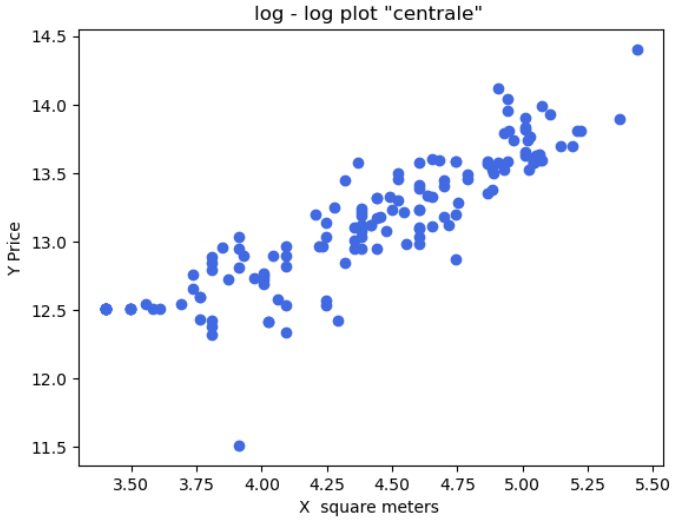
1. Square Meters: I initiated the exploration by plotting home prices against square meters, as it seemed to be the covariate with the most significant impact on prices. After experimenting with the data, I observed that applying the logarithm to both variables helped homogenize the variance.



2. Condominium Fees: Similar to square meters, applying the logarithm to condominium fees also aided in homogenizing the variance.

3. Zone Variations: Upon further examination, I noticed that the effect of square meters on prices varied across different zones. To capture this variation, I generated interaction dummies between zone dummies and square meters.

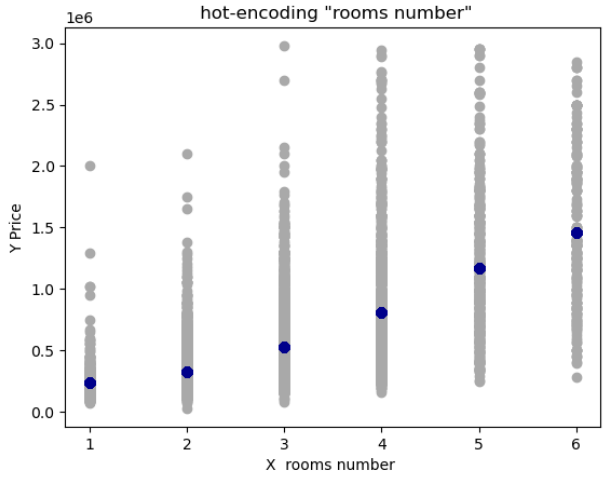


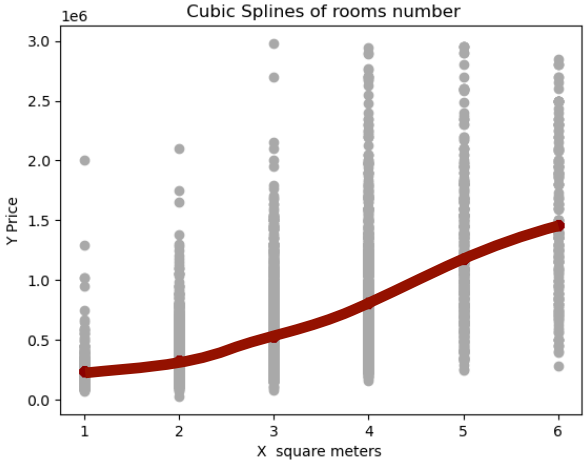


4. Exploration of Other Variables: Subsequently, I delved into analyzing the remaining variables. I developed two functions: one for generating dummy variables to hot-encode categorical variables, and another for creating splines to capture non-linear relationships.

For instance, I visualized the relationship between 'room number' and prices using both dummy variables and splines to account for potential non-linearities.

By systematically exploring the relationships between covariates and home prices, I aimed to gain insights into the underlying patterns and dynamics of the housing market in Milano.





Part 4: Building the First Model

To assess the predictive performance of the dataset, I began by implementing a linear regression model using the 'Sklearn' package. This allowed me to identify significant

variables and determine whether to utilize splines or hot-encode certain variables based on cross-validation results.

1. Linear Regression: Using the `'GridSearchCV()'` function from 'Sklearn', I conducted a 10-fold cross-validation to optimize the predictors (maainly choosing between slines and dummies). Additionally, I employed Lasso regression for variable selection and regularization to control variance. Prior to this, standardization of variables was performed using `'StandardScaler()'` from 'Sklearn'. Calibration of the 'lambda' parameter for Lasso was achieved through 10-fold cross-validation.

2. Ridge Regression: Similarly, a Ridge regression model was implemented following standardization of variables. Cross-validation was again used to calibrate the 'lambda' parameter for the Ridge. Comparison of performance between Lasso and Ridge models using cross-validation indicated that Ridge outperformed Lasso, leading to its selection as the preferred model.

3. Variable Selection: During model building, I observed that some variables created from the 'other features' category had very few observations. Employing cross-validation, I identified and removed such variables to streamline the model and improve its performance.

By systematically iterating through model building steps and leveraging cross-validation techniques, I aimed to develop a robust predictive model for home prices in Milano.

Part 5: Implementing the Second Model

Recognizing the need for improved model performance, I opted to explore a different approach and implemented X Gradient Boosting from ‘xgboost’, a highly regarded model known for its effectiveness in predictive tasks.

1. Data Preprocessing: Before fitting the X Gradient Boosting model, I undertook significant modifications to the dataset. This included removing interaction variables, splines, and dummy variables. Categorical variables were appropriately encoded using the .astype('category') method to prepare the data for the model.

2. Model Development: With the dataset prepared, I proceeded to implement the X Gradient Boosting model. To identify the optimal parameters for the model, I employed 10-fold cross-validation, leveraging the same package utilized for Ridge regression. Through this process, I determined the following optimal parameter values:

1.’max_depth': 6, setting the maximum depth of each tree.

'learning_rate': 0.025, controlling the contribution of each tree to the final prediction.

'n_estimators': 2100, specifying the number of trees in the ensemble.

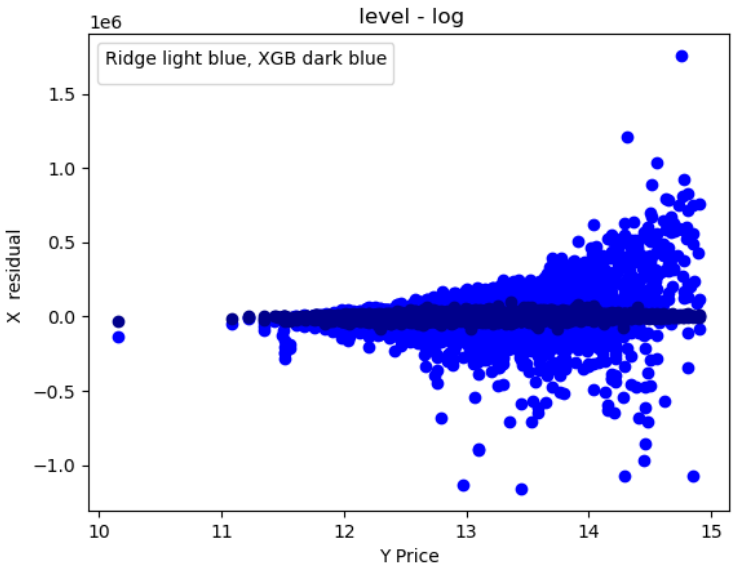
'lambda' and 'alpha': Both set to 0, indicating no regularization to prevent overfitting, thereby making the more model similar to normal gradient boosting.

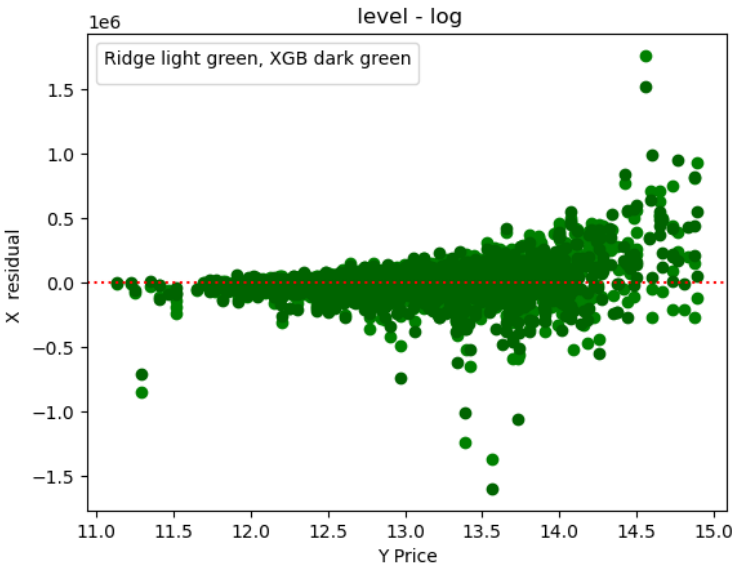
'subsample': 0.7, determining the proportion of data randomly sampled for training each tree.

By fine-tuning these parameters through cross-validation, I aimed to optimize the performance of the X Gradient Boosting model and enhance its predictive accuracy for home prices in Milano.

Part 6: Stacking & Averaging

After observing that both the X Gradient Boosting (Xgb) and Ridge models exhibited similar performances, with Xgb slightly outperforming Ridge in terms of Mean Absolute Error (MAE), I proceeded to examine the residuals of both models.





1. Residual Analysis: Plotting the residuals of the test data (2000 out of 8000 observations used for testing) in green and the training data (6000 observations out of 8000 used for training) in blue, revealed that each model captured different aspects of the data, despite their similar overall performance.

2. Stacking Approach: Given the possible complementary nature of the two models, I opted to combine them. To do so, I stacked the predictions generated by each model, using the 2000 testing observations in a Ridge model, alongside the other covariates used for the Ridge estimation. Through cross-validation, I attempted to estimate the lambda parameter for Ridge.

3. Observations and Insights: During this process, I noted that the lambda parameter was excessively large, implying that Ridge was shrinking most variables. Consequently, only the predictions from the two models remained significant.

4. Weighted Average: Subsequently, I conducted a linear regression using only the predictions from both models as independent variables and the actual prices as the dependent variable. The coefficients obtained from this regression revealed the optimal combination of the two models. I determined that a weighted average, with the coefficients from the regression serving as weights, yielded the best results.

	coef	std err	t	P>|t|	[0.025	0.975]
Ridge Predictions	0.3379	0.036	9.269	0.000	0.266	0.409
XGB Predictions	0.6610	0.036	18.151	0.000	0.590	0.732

5. Model Training and Prediction: Finally, I trained both models using the entire dataset, computed a weighted average of their prediction vectors using the regression coefficients, and exported the results.

Part 7: Inference

For this section, I simplified the dataset used for Ridge regression, primarily considering most variables as continuous and retaining dummies only for the zones for more clearness and interpretability.

1. Variable Analysis: Starting from the top, I observed that while availability displays a negative correlation with price, the coefficient is small and statistically insignificant, indicating it doesn't play a significant role in price prediction.

2. Significant Variables: Both the number of bathrooms and rooms exhibit positive correlations with price, supported by high t-scores, indicating their significance in predicting prices. Similarly, the condition of the house, whether new or refurbished, significantly impacts prices.

R-squared:	0.734
Adj. R-squared:	0.724

	coef	std err	t	P>|t|	[0.025	0.975]
availability	-8.797e-05	0.000	-0.276	0.782	-0.001	0.001
bathrooms_number	0.1599	0.010	16.692	0.000	0.141	0.179
conditions	0.0784	0.006	13.829	0.000	0.067	0.090
condominium_fees	0.0653	0.008	8.187	0.000	0.050	0.081
energy_efficiency_class	0.0171	0.003	6.706	0.000	0.012	0.022
floor	0.0218	0.002	9.785	0.000	0.017	0.026
heating_centralized	0.0006	0.011	0.057	0.955	-0.020	0.022
lift	0.0684	0.012	5.603	0.000	0.044	0.092
pool	0.1561	0.041	3.846	0.000	0.077	0.236
private_parking	0.0405	0.010	4.107	0.000	0.021	0.060
rooms_number	0.1663	0.007	24.954	0.000	0.153	0.179
shared_parking	0.0149	0.010	1.473	0.141	-0.005	0.035
square_meters	2.9860	0.033	91.380	0.000	2.922	3.050
terrace	0.1143	0.012	9.859	0.000	0.092	0.137
total_floors_in_building	-0.0065	0.002	-3.178	0.001	-0.011	-0.002
condominium_fees_dummy	-0.3439	0.039	-8.753	0.000	-0.421	-0.267
zone_affori	10.9899	0.250	43.949	0.000	10.500	11.480
zone_amendola - buonarroti	10.0603	0.566	17.782	0.000	8.951	11.169
zone_arco della pace	10.3859	0.425	24.445	0.000	9.553	11.219
zone_arena	11.5022	0.457	25.160	0.000	10.606	12.398
zone_argonne - corsica	10.7417	0.292	36.741	0.000	10.169	11.315
zone_afforiINT	-2.9608	0.071	-41.768	0.000	-3.100	-2.822
zone_amendola - buonarrotiINT	-2.5344	0.127	-19.884	0.000	-2.784	-2.285
zone_arco della paceINT	-2.6101	0.110	-23.787	0.000	-2.825	-2.395
zone_arenaINT	-2.8694	0.115	-24.907	0.000	-3.095	-2.644
zone_argonne - corsicaINT	-2.7887	0.080	-34.971	0.000	-2.945	-2.632

3. Condominium Fees: Condominium fees, despite numerous null values, display a negative correlation with price for non-null observations. This effect is captured by a dummy variable, indicating a decrease in price with higher fees.

4. Parking and Building Features: Having more private parking spots, a lift, or being on a higher floor positively influences price. Conversely, the significance of shared parking is comparatively lower.

5. Heating System: The type of heating system, whether centralized or not, does not significantly affect house prices based on the available data.

6. Luxury Features: Luxury amenities such as a pool or terrace significantly increase the value of a property, as evidenced by their positive coefficients and significance.

7. Square Meters and Zone Dummies: Square meters and zone dummies are crucial variables affecting house prices. The plots showed earlier reveal varying price trends across different zones, with a linear — or similar — relationship between size and price within each zone. The introduction of interaction coefficients further underscores the significance of these variables, as demonstrated by their significant coefficients in the plots.

Final remark: In light of the fact that prices have been logged, it's essential to note that the coefficients now represent the percentage increase in prices for each unitary increase in the variable. Conversely, for logged independent variables, as **square meters**, the coefficients indicate the percentage increase in prices for every unitary percentage increase in the variable. This distinction is fundamental for accurately understanding the impact of each variable on house prices.

Note:

The Jupyter lab notebook contains all the code for generating the results. To do so it is sufficient to run all the cells in order.

In the subsection “splines and dummies”, can be found the functions to generate splines and dummies. In particular for splines used the patsy method with the `bs( )` function:

```
“patsy.dmatrix(f”bs(dataframe[name], df={df}, degree=3,
include_intercept=False) - 1", {"x": dataframe[name]}”
```

While for dummies used pandas:

```
pd.get_dummies(df[name], prefix=name).astype(int)
```